VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Animované prezentace linkových kódů v prostředí PGF/TikZ

Animated Presentations of Line Codes Using PGF/TikZ Package

2019

Tomáš Hmilanský

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Tomáš Hmilanský**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2601R013 Telekomunikační technika

Téma: Animované prezentace linkových kódů v prostředí PGF/TikZ
Animated Presentations of Line Codes Using PGF/TikZ Package

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je připravit soubor animací prezentujících principy linkových kódů spustitelných v souboru PDF.

1. Popište v krátkosti prostředí PGF/TikZ s důrazem na tvorbu animací s funkčními příklady.
2. Popište (po konzultaci s vedoucím práce) principy vybraných linkových kódů a jejich klíčové vlastnosti.
3. Vytvořte animace prezentující princip linkových kódů.
4. Otestujte možnosti začlenění vytvořených animací do souboru PDF s využitím prostředí TeXLive.

Seznam doporučené odborné literatury:

[1] TikZ manual. URL: http://www.texample.net/media/pgf/builds/pgfmanualCVS2012-11-04.pdf

[2] MACHÁČEK, Pavel. *Animované prezentace analogových a digitálních modulací v prostředí PGF/TikZ.* Ostrava, 2016. Bakalářská práce. Vysoká škola báňská - Technická univerzita Ostrava. Fakulta elektrotechniky a informatiky. Vedoucí práce Skapa, Jan. URL: http://hdl.handle.net/10084/116089

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Skapa, Ph.D.**

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2019

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

prof. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. Dubna 2019

.....

Rád bych na tomto místě poděkoval vedoucímu bakalářské práce Ing. Janu Skapovi, Ph.D. za odborné vedení a vstřícný přístup, dále bych rád poděkoval mé rodině za zázemí a morální podporu, protože bez nich by tato práce nevznikla.

Abstrakt

Hlavním cílem této bakalářské práce je vytvoření souboru animací vybraných linkových kódů, které jsou využívány při přenosu digitálních signálů. Práce obsahuje teoretickou a praktickou část. Teoretická část pojednává o prostředí typografického programu TEX a jeho souboru maker LATEX s hlavním zaměřením na makra PGF a TikZ. Dále popisuje základní principy linkových kódů a jejich klíčové vlastnosti. Výstupem praktické části je soubor animací jednotlivých linkových kódů spustitelných v programech podporujících formát PDF. Práce může také posloužit jako manuál pro práci s makry PGF a TikZ.

Klíčová slova: animace, linkové kódy, LATEX , PGF, TikZ

Abstract

The main aim of this bachelor thesis is to create a set of animations of selected line codes, which are used in the transmission of digital signals. The thesis contains a theoretical and practical part. The theoretical part deals with the TEX typographic program environment and its LATEX package set with the main focus on PGF and TikZ packages. It also describes the basic principles of line codes and their key features. The output of the practical part is a set of animations of individual line codes executable in software supporting PDF format. Thesis can also serve as a manual for working with PGF and TikZ macros.

Key Words: animation, line codes, LATEX , PGF, TikZ

Obsah

Seznam použitých zkratek a symbolů

TEX	–	software pro sazbu textu
LATEX	–	balík maker programu TeX
PGF	–	Portable Graphics Format
TikZ	–	TikZ ist kein Zeichenprogramm
PDF	–	Portable Document Format
SVG	–	Scalable Vector Graphics
WYSIWYG	–	What You See Is What You Get
SWF	–	Shockwave Flash
fps	–	Frames Per Second
RZ	–	Return to Zero
NRZ	–	Non-Return to Zero
AMI	–	Alternate Mark Inversion
2B1Q	–	Two-Binary, One-Quaternary
QPSK	–	Quadrature Phase-Shift Keying
ISDN	–	Integrated Services Digital Network

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

Úvod

Digitální přenos informací je nedílnou součástí každodenního života a využíváme ho, aniž bychom si to uvědomovali. Ale aby bylo možné jakkoliv přenášet digitální signál, je nutné jej nějak upravit. V tom případě rozlišujeme přenos v základním a přeloženém pásmu. V přeloženém pásmu pro přenos využíváme modulace signálu, kdežto v základním pásmu se využívají linkové kódy. Cílem bakalářské práce je vytvoření sady animací představující principy linkových kódů.

V prvních dvou kapitolách se budu věnovat typografickému prostředí, v kterém budu animace vybraných linkových kódů vytvářet. Pracovat budu v prostředí sázecího programu TEX, a to s jeho souborem maker LATEX. Ten využívá předdefinované vzhledy dokumentů a formátovací příkazy pro sazbu textu, proto je dokument nutné před výsledným náhledem přeložit. LATEX mimo jiné obsahuje makra PGF a TikZ, které umožňují vysázet vektorovou grafiku pomocí algebraického popisu. Díky těmto dvěma makrům lze vytvářet mimo jiné i animace spustitelné v PDF souborech. Animace obsahují uživatelské rozhraní, přes které můžeme animace ovládat. V uživatelském rozhraní nalezneme tlačítka pro spuštění nebo pozastavení, krokování, zrychlení a zpomalení. Proto se vytvořené animace hodí zejména jako výukové materiály. Tyto dvě kapitoly by měly posloužit jako návod, jak pracovat v prostředí LATEXu a jak v něm vytvářet animace.

V třetí kapitole popíšu jaké rozlišujeme linkové kódy a následně popíšu principy vybraných kódů a jejich hlavní vlastnosti. Jak bylo výše zmíněno, linkové kódy se využívají pro digitální přenos přes přenosové médium. Lze je využít pro metalická vedení, ale i pro optické médium nebo volný prostor. Jedná se vlastně o libovolnou reprezentaci jednotlivých bitů digitálního signálu v přenosovém médiu tak, aby je bylo možné při obdržení na přijímací straně dekódovat. Slouží hlavně k odstranění stejnosměrné složky signálu, kterou nelze přenášet, a zajišťují synchronizaci mezi vysílačem a přijímačem.

Následující kapitola bude věnována praktické části bakalářské práce. V rámci této kapitoly vytvořím soubor animací vybraných linkových kódů se stručným popisem vytváření a zdrojovým kódem animací. Animace jsou jasně srozumitelné a principy linkových kódů jsou z nich snadno pochopitelné. Jelikož animace obsahují uživatelské rozhraní, tak mohou posloužit jako výukové materiály například pro předmět "Přenosové systémy a média". V poslední kapitole popíšu možnosti spuštění animací v různých programech podporujících formát PDF. Celá práce je vytvořena v typografickém prostředí LATEX a to včetně textu i animací. Využiji volně dostupného programu MiKTeX 2.9 pro operační systém Windows, který obsahuje všechna makra LATEXu a také editor TeXworks pro vytváření a editaci TeXových dokumentů. Všechny animace naleznete v přílohách práce.

1 Základy LaTeXu

LaTeX je volně dostupný software pro tvorbu profesionálně vypadajících dokumentů. Proto je vhodný zejména k psaní knih, odborných článků nebo bakalářských či diplomových prací. Obsahuje velké množství maker neboli balíčků, které implementují sady příkazů pro vytváření nejrůznějších objektů, například:

- tvorba tabulek - balíček array
- tvorba matematických konstrukcí - balíček mathtools
- tvorba vektorové grafiky - balíčky PGF a TikZ

Narozdíl od programu Microsoft Word se v LaTeXu k formátování obsahu používá nízkoúrovňový programovací jazyk TeX. Kvůli tomu je nutné pro vytvoření například souboru PDF nejdříve dokument zkompilovat a poté exportovat v daném formátu. Na začátku je vždy potřeba objasnit, o jaký dokument jde a uvést jeho klíčové vlastnosti. Následně se naimportují potřebné balíčky a poté se uvede začátek dokumentu. Nakonec se uvede také konec dokumentu.[1] Příklad, jak vytvořit jednoduchý dokument, můžete vidět na následujícím kódu:

```
% typ dokumentu a jeho vlastnosti (například velikost písma či papíru, při více
    vlastnostech se každá jednotlivě oddělí čárkou)
\documentclass[<vlastnosti>]{<typ dokumentu>}

% Import použitého balíčku, za <makro> se doplní název balíčku
\usepackage{<makro>}

\begin{document} % začátek dokumentu

% Zde se píše požadovaný obsah dokumentu.

\end{document} % konec dokumentu
```

Výpis 1: Ukázka formátu základního dokumentu

2 PGF/TikZ

2.1 PGF

PGF je jeden z mnoha balíčků LaTeXu, který umožňuje vytváření jednoduché vektorové grafiky a je využíván balíčkem TikZ. Jak už bylo řečeno, TeX je nižší programovací jazyk. Taktéž je tomu s balíčkem PGF. Nejedná se tedy o WYSIWYG, což v překladu znamená "co vidíš, to dostaneš", proto nelze vidět jak grafika vypadá, aniž bychom kód zkompilovali. Ač je tedy vytváření jednoduché grafiky velice rychlé a umožňuje využívání ostatních maker, tak je obtížnější si představit, co kód reprezentuje. Další nevýhodou je rekompilace i po malé úpravě kódu, která chvíli trvá.

PGF se zkládá ze tří vrstev:

Systémová vrstva

Jedná se o nejnižší vrstvu makra PGF. Uživatel s touto vrstvou vůbec nepracuje, alespoň ne přímo. Tato vrstva interpretuje děj, který se odehrává v ovladači. Překládá systémové příkazy, které zadáváme do vývojového prostředí, na posloupnost několika jiných speciálních příkazů (příkazy `\special`). Pomocí těchto speciálních příkazů se následně generuje výsledná grafika, převoditelná do formátu PDF.

Základní vrstva

Základní vrstva se skládá z jádra a z přídavných modulů, které rozšiřují jádro o několik speciálně určených příkazů. Jádro se dále skládá z několika nezávislých balíčků. Vrstva obsahuje mnohem jednodušší příkazy k vytváření grafiky než vrstva systémová. Například kdybychom chtěli pomocí systémové vrstvy vytvořit nějaký geometrický útvar, museli bychom napsat poměrně dlouhý kód. Základní vrstva pro vytvoření geometrického obrazce naproti tomu využije mnohem méně příkazů.

Aplikační vrstva

Aplikačních vrstev může být více a jedná se o soubor příkazů nebo speciální syntaxe. Díky tomu je využívání základní vrstvy jednodušší. Kód základní vrstvy totiž bývá občas příliš ukecaný. Například pokud bychom chtěli vykreslit geometrický obrazec, tak u základní vrstvy využijeme sice méně příkazů než na vrstvě systémové, ale pořád jich pár bude. Aplikační vrstva ale pro vykreslení obrazce využije jen jeden příkaz. Pro ovládání makra PGF využije uživatel právě příkazy aplikační vrstvy, popřípadě pár příkazů základní vrstvy.[2][3]

2.2 TikZ

Narozdíl od PGF a TeXu je TikZ soubor vysokoúrovňových maker a v překladu tato zkratka znamená "TikZ není kreslící program". Pro využívání tohoto makra je nutné ho na začátku dokumentu naimportovat a blok příkazů musí být uvnitř příkazů pro začátek a konec:

```
\usepackage{tikz}

\begin{document}
\begin{tikzpicture} % Začátek TikZ příkazů

% Blok příkazů

\end{tikzpicture} % Konec TikZ příkazů
\end{document}
```

Výpis 2: Ukázka využití makra TikZ

Za příkaz pro začátek TikZu můžeme ještě přidat vlastnosti pro následný blok příkazů do hranatých závorek. PGF se neimportuje, jelikož jsou tyto dva balíčky velice úzce spjaté (TikZ využívá PGF). TikZ je inspirovaný jinými jazyky jako METAFONT, balíkem maker PSTRICKS nebo grafikou SVG.[3] Skládá se z několika designových principů:

- speciální syntaxe pro definování bodů
- speciální syntaxe pro definování přímek
- volby pro přímky
- mapovací syntaxe pro grafické parametry
- speciální syntaxe pro grafy
- speciální syntaxe pro uzly v grafu
- shlukování grafických parametrů
- koordinace systému transformace[3]

Jedním z nejdůležitějších příkazů z prostředí TikZ je příkaz \draw. Kdybychom si pro příklad chtěli vykreslit jednoduchý červený obdélník, použitý výraz by vypadal takto:

```
\begin{tikzpicture}

% V hranatých závorkách se uvádí vlastnosti obrazce, následuje umístění, typ
    obrazce a velikost
\draw[red] (0,0) rectangle (1,2);

\end{tikzpicture}
```

Výpis 3: Vykreslení červeného obdélníku

Výsledný obdélník:

Obrázek 1: Vykreslený obdélník pomocí TikZu

2.3 Vytváření grafů

Pro interpretaci průběhu linkových kódů využijeme balíčku pgfplots, pomocí kterého budeme moci vykreslovat grafy. Pgfplots nabízí širokou škálu možností, jak jednoduše vykreslovat grafy, například umožňuje:

- tvorbu normálních grafů
- tvorbu logaritmických grafů
- tvorbu 2-rozměrných grafů
- tvorbu 3-rozměrných grafů
- jednoduchý popis os
- popis jednotlivých bodů v grafu[4]

Jelikož se jedná o balíček, tak musí být naimportován na začátku dokumentu a také musí být naznačený začátek a konec prostředí pro vytvoření grafu (stejně jako při vytváření obrazců, jelikož se jedná o makra PGF/TikZ, tak se využivá prostředí tikzpicture). Pro vykreslení grafu musíme však navíc přidat začátek a konec s klíčovým slovíčkem axis. Za začátek grafu definujeme v hranatých závorkách parametry grafu a jeho os (nejsou však povinné), například:

- název grafu - name
- umístění grafu - at
- šířka grafu - width
- výška grafu - height
- název osy x a y - xlabel a ylabel
- minima a maxima os - xmin, xmax, ymin a ymax

- zobrazení mřížky - grid[4]

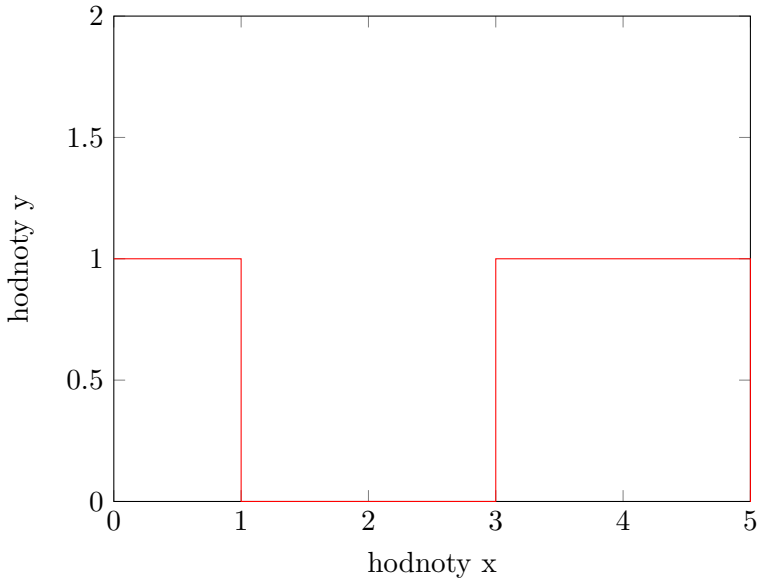
Následný graf se do obrázku přidává pomocí příkazu \addplot a v případě více funkcí v jednom grafu využijeme příkaz \legend pro pojmenování jednotlivých křivek. Pokud bychom chtěli vykreslit více grafů se stejnými vlastnostmi, tak balíček pgfplots obsahuje knihovnu groupplots, která se nám o to postará. S knihovnou se nakládá stejně, jak tomu jsme zvyklí, tzn. musí být na začátku naimportována, musí být uveden začátek a konec bloku všech souvisejících grafů a každý jednotlivý graf musí být oddělen příkazem \nextgroupplot. V grafu pak můžeme vykreslit buď matematické funkce nebo můžeme zadat jednotlivé hodnoty x a y (ty lze zadat buď přímo do kódu nebo v textovém souboru, který je pak naimportován).

```
\usepackage{tikz}
\usepackage{pgfplots}

\begin{tikzpicture}
\begin{axis}[
name = Data, width = 10cm, height = 5cm, xlabel = {hodnoty x}, ylabel = {
    hodnoty y}, xmin = 0, xmax = 5, ymin = 0, ymax = 1
] % Začátek prostředí axis [parametry grafu]
% Přidání grafu [parametry křivky], coordinates - hodnoty grafu ve tvaru (x,y)
\addplot[thick, color = red] coordinates{
(0,1)
(1,1)
(1,0)
(2,0)
(3,0)
(3,1)
(4,1)
(5,1)
};
\end{axis}
\end{tikzpicture}
```

Výpis 4: Tvorba grafů pomocí pgfplots

Výsledný graf:



Obrázek 2: Vykreslený graf pomocí pgfplots

2.4 Vytváření animací

Pro vytváření samotných animací máme dvě varianty, a to buď budeme pracovat s balíčky ifthen a animate nebo jen s balíčkem animate a jeho příkazem `\multiframe`. Balíček ifthen nám umožní využívat podmínky v prostředí TeXu a také základní smyčku whiledo. Podmínky jsou využívany snad v každém programovacím jazyku a slouží pro určení bloku kódů, který se vykoná při splnění či nesplnění dané podmínky. Naproti tomu příkaz `\multiframe` se chová podobně jako smyčka, proto bychom v tomto případě ifthen vůbec nepotřebovali. Balíček animate pak obsahuje rozhraní pro vytváření animací spustitelných v souborech PDF. Nejedná se tedy o animace ve video formátu, ale o animace vektorové grafiky podobné jako SWF formát.

2.4.1 Balíček ifthen

Stejně jako ostatní balíčky i ifthen musí být na začátku dokumentu naimportován. Z balíčku využijeme příkazy `\ifthenelse` a `\whiledo`. Nejdříve si ale musíme nadefinovat proměnnou, pomocí které budeme provádět jednotlivé iterace cyklu, popřípadě kterou budeme ověřovat v podmínce. TeX v tomto případě používá proměnnou counter a jedná se o datový typ long integer. Jak se s proměnnou pracuje, můžete vidět na následující ukázce:

```
\newcounter{name} % Deklarace proměnné, name = název proměnné
\setcounter{name}{value} % Nastavení hodnoty, value = požadovaná hodnota
\stepcounter{name} % Zvýšení proměnné o 1
\addtocounter{name}{value} % Zvýšení proměnné o určitou hodnotu
```

Výpis 5: Práce s proměnnou counter

Struktura podmínky ifthenelse obsahuje celkem tři bloky, umístěné za příkazem a každý je oddělen složenými závorkami. První blok obsahuje podmínku samotnou (například x < 5). Druhý blok obsahuje sadu příkazů, která se provede, pokud byla podmínka splněna a pokud podmínka nebyla splněna, tak se provede sada příkazů z třetího bloku. Cyklus whiledo má oproti tomu za příkazem bloky pouze dva. V prvním je uvedena podmínka, určující kolikrát se cyklus zopakuje a v druhém je sada příkazů, která se bude po dobu cyklu opakovaně provádět.[5] Jak ifthenelse a whiledo používat, můžete vidět na následující ukázce:

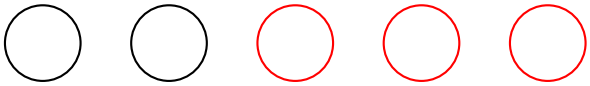
```
\usepackage{ifthen}
\usepackage{tikz}

\newcounter{x}
\setcounter{x}{1}

% Cyklus whiledo proběhne 5x (\thex značí proměnnou x)
\whiledo{\thex < 6}{
% Pokud platí x < 5, tak se vykreslí černý kruh, pokud neplatí, tak se vykreslí
    červený kruh
  \ifthenelse{\thex < 3}{
    \begin{tikzpicture}
    \draw[black,thick] (\thex,0) circle (0.5);
    \end{tikzpicture}
  }{
    \begin{tikzpicture}
    \draw[red,thick] (\thex,0) circle (0.5);
    \end{tikzpicture}
  }
  \stepcounter{x}
}
```

Výpis 6: Práce s cykly a podmínkami

Výstup kódu:



Obrázek 3: Vykreslení kruhů pomocí TikZu a smyčky whiledo

2.4.2 Multiframe

Příkaz \multiframe má jednu velkou výhodu oproti využívání podmínek a cyklů, a to že dokáže zpracovat čísla s desetinnou čárkou. Jeho syntaxe pak obsahuje celkem tři bloky. V prvním bloku uvádíme, kolik bude celkově snímků v dané animaci, v druhém bloku se pak uvádí proměnná a její krok, např. ix=0+1, kdy 0 nám udává počáteční hodnotu a +1 krok.[6] Zápisem ix si navíc volíme, jaký typ čísel budeme používat:

- ix - integer proměnné
- rx - reálné proměnné
- dx - proměnné typu rozměr

2.4.3 Balíček animate

Díky balíčku animate jsme schopni vytvořit animace vektorové grafiky, které budou spustitelné v PDF souborech. Rozhraní animací však využívá javascript, proto je lze spustit jen v softwaru pro čtení PDF, který javascript podporuje. Animate nám dovoluje vytvářet animace dvojím způsobem. V rámci bakalářské práce využijeme metodu animateinline. Blok animace musí být, jak jsme již zvyklí, ohraničen příkazem pro začátek a konec daného prostředí. K příkazu pro začátek prostředí můžeme přidat do hranatých závorek parametry animace a do dalších hranatých závorek počet snímků za sekundu. Hodnotu fps si volíme podle sebe a parametry můžeme uvést například následující:

- poster - nastavení počátečního obrázku
- autoplay - automatické spuštění animace
- controls - ovládání ovládacího rozhraní animace
- palindrome - po skončení se animace přehraje pozpátku
- loop - animace se cyklicky opakuje
- step - animace se krokuje klikáním myši[6]

Aby animace v PDF souboru pracovala jak má, tak je nutné TeXovský dokument přeložit dvakrát. Prvním překladem se vytvoří samotný dokument a animace a po druhém přeložení se zpřístupní ovládací prvky animace. Nesmíme zapomenout, že pro používání balíčku jej musíme na začátku dokumentu naimportovat. Příklad jednoduché animace můžete vidět na následujícím kódu:

```
\newcounter{x}
\setcounter{x}{0}
\begin{animateinline}[poster = last, controls]{8}
\whiledo{\thex < 11}{
   \begin{tikzpicture}
   \begin{axis}[
   name = Data, width = 10cm, height = 5cm, xlabel = {hodnoty x}, ylabel = {
       hodnoty y}, xmin = 0, xmax = 5, ymin = 0, ymax = 1]
   \addplot[thick, color = red, restrict x to domain=0:\thex] coordinates{
   (0,0)
   (0,1)
      .
      .
   (6,0)};
   \end{axis}
   \end{tikzpicture}
   \stepcounter{x}
   \ifthenelse{\thex < 11}{
      \newframe}{
      \end{animateinline}}}
```

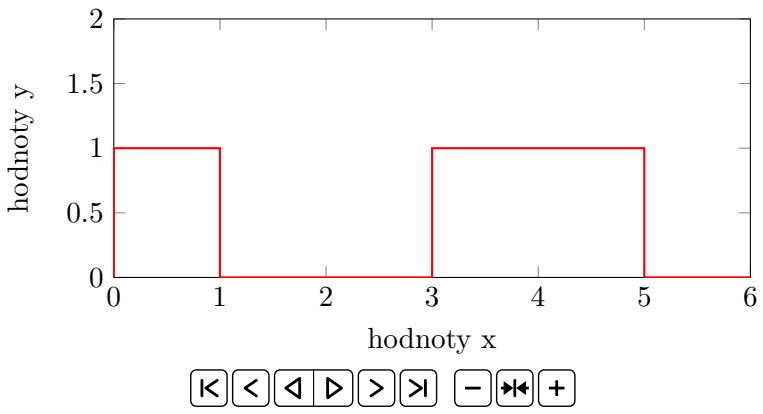
Výpis 7: Vytvoření jednoduché animace pomocí balíčku ifthen

Pokud bychom využili příkaz \multiframe, kód by vypadal následovně:

```
\begin{animateinline}[poster = last, controls]{8}
\multiframe{11}{ix=0+1}{
   \begin{tikzpicture}
   \begin{axis}[
   name = Data, width = 10cm, height = 5cm, xlabel = {hodnoty x}, ylabel = {
       hodnoty y}, xmin = 0, xmax = 5, ymin = 0, ymax = 1]
   \addplot[thick, color = red, restrict x to domain=0:\ix] coordinates{
   (0,0)
   (0,1)
      .
      .
   (6,0)};
   \end{axis}
   \end{tikzpicture}
}
\end{animateinline}
```

Výpis 8: Vytvoření jednoduché animace pomocí příkazu multiframe

Výsledná animace:



Obrázek 4: Ukázka jednoduché animace

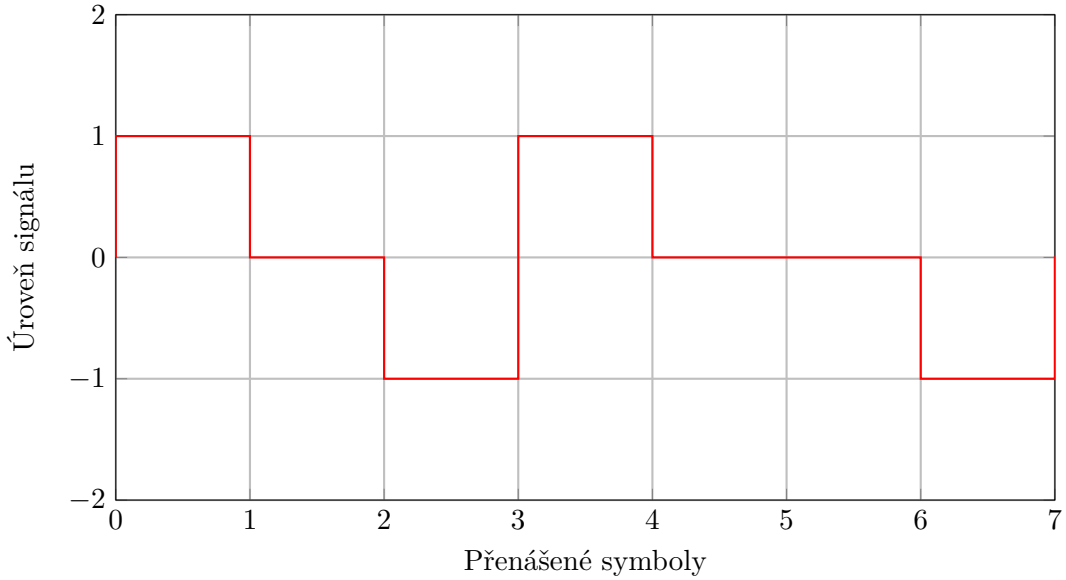
3 Linkové kódy

Linkové kódy jsou nedílnou součastí přenosu digitálního signálu. Využíváme je k přenosu informací v základním pásmu, do kterého spadají signály o nižší frekvenci. Existuje mnoho různých linkových kódů pro různá využití. Jejich hlavním cílem je dosáhnutí co možná nejvyšší přenosové rychlosti, odstranění stejnosměrné složky signálu a zajištění synchronizace příjmače s vysílačem. Lze je dělit dle různých kritérií:

- podle polarity:
  - unipolární - nabývají pouze kladných hodnot
  - polární - nabývají kladných i záporných hodnot
  - bipolární - nabývají kladných i záporných hodnot a nuly
- podle návratu k nule:
  - s návratem k nule (RZ) - signál se během jednoho impulzu vrací k nule
  - bez návratu k nule (NRZ) - signál se během jednoho impulzu k nule nevrací
- podle počtu úrovní:
  - dvojúrovňové
  - trojúrovňové
  - vícerovňové

3.1 AMI

Jedná se o trojúrovňový, bipolární linkový kód bez návratu k nule. AMI je synchronní kódovací technika a výsledný signál je bipolární. Tento linkový kód byl hojně využíván v první generaci PCM sítí. Hlavní výhodou linkového kódu AMI je odstranění stejnosměrné složky signálu, díky tomu můžeme navíc přenášet elektrický proud, který může být následovně využitý k napájení přístrojů na trase přenosu (např. opakovače). Kódování je následovné, nuly jsou kódovány jako nulový signál a jedničky se střídavě kódují jako kladný a záporný signál.[7] Například pokud bychom chtěli pomocí AMI zakódovat posloupnost bitů 1011001, výsledný signál by vypadal takto:



Obrázek 5: Ukázka kódování AMI (zákódována sekvence 1011001)

3.2 Manchester

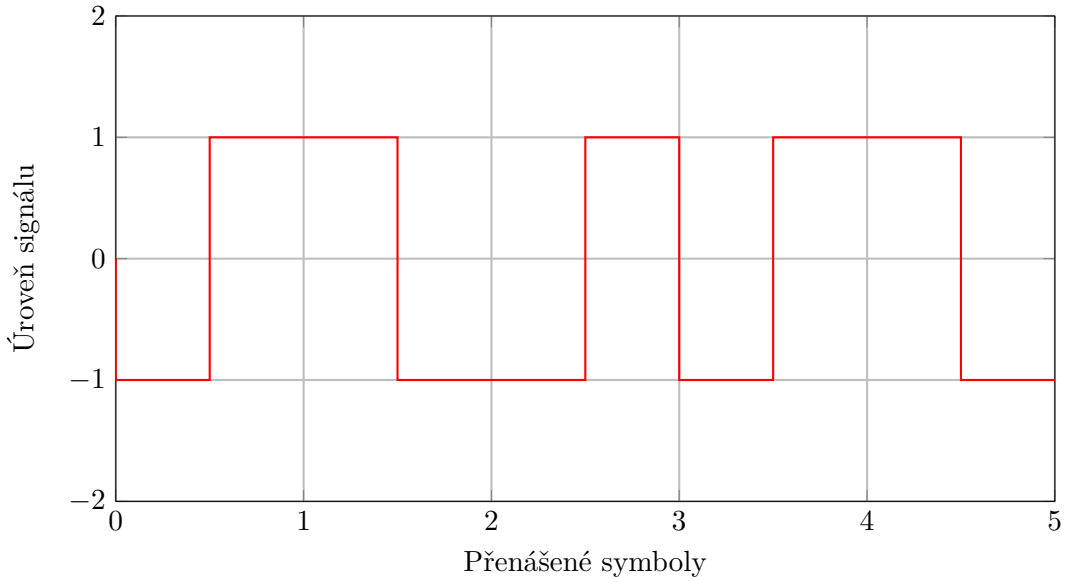
Manchester je dvojúrovňový, polární linkový kód s návratem k nule, který využívala například technologie ethernet. Jako reprezentace bitů zprávy se v polovině intervalu pro jeden bit přechází z jedné úrovně do druhé, přičemž rozlišujeme dvě možnosti jak daný přechod realizovat. První způsob publikoval již v roce 1949 G. E. Thomas, kdy při kódování nuly se přechází z nižší úrovně na vyšší a naopak tomu je pro jedničku. Druhý způsob je popsán standartem IEEE 802.3, kdy při kódování nuly se přechází z vyšší úrovně na nižší a naopak tomu zase je pro jedničku. Klíčovými vlastnostmi je absence stejnosměrné složky signálu a dobrá synchronizace mezi vysílačem a příjímačem, a to právě díky přechodu mezi úrovněmi v půlce intervalu jednoho bitu.[8]

3.3 Diferenciální Manchester

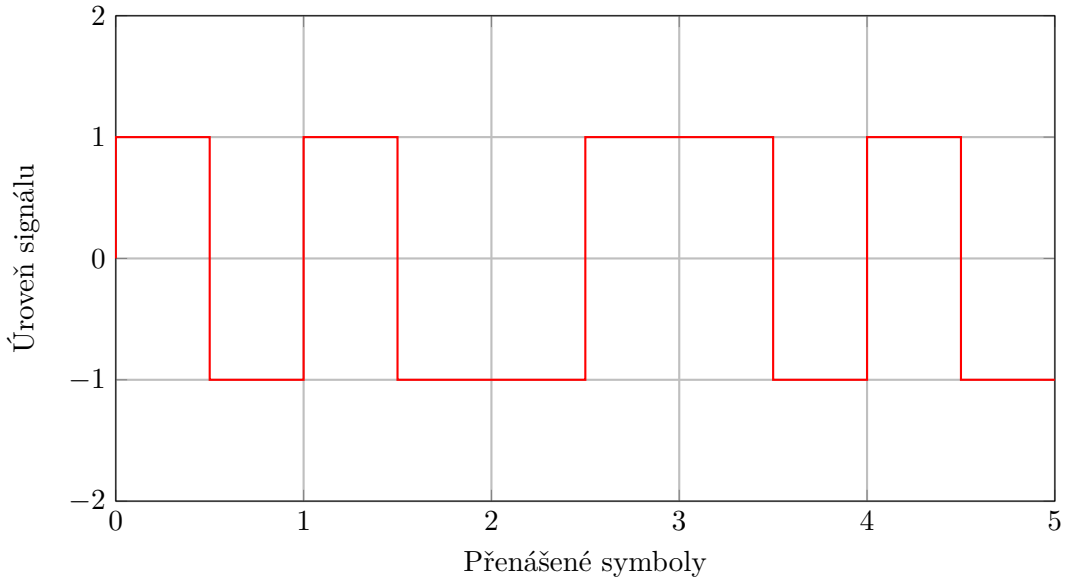
Diferenciální Manchester je stejně jako Manchester dvojúrovňový, polární linkový kód s návratem k nule, využívaný například v technologii Token Ring. Samotné kódování spočívá v absenci nebo přítomnosti přechodu mezi úrovněmi po přenesení bitu. Máme dvě možnosti, jak bity zakódovat, a to buď přechodem při nule nebo přechodem při jedničce. Například pokud budeme kódovat přechodem při jedničce, tak při přivedení nuly na vstup je výstupní úroveň stejná jako u předchozího bitu a ke změně úrovně dochází stejně jako u Manchesteru v půlce intervalu přenášeného bitu. Pokud však přivedeme na vstup jedničku, tak se na začátku úroveň změní

a stejně tak se změní poté v půlce intervalu. Hlavní výhodou je dobrá synchronizace vysílače s příjímačem.

Rozdíly mezi kódováním Manchester a Diferenciální Manchester můžete vidět na následujících obrázcích:



Obrázek 6: Ukázka kódování Manchester podle standardu IEEE 802.3 (zákódována sekvence 10110)



Obrázek 7: Ukázka kódování Diferenciální Manchester (zákódována sekvence 10110)

3.4 HDB3

Linkový kód HDB3 je odvozený z kódování AMI, které vlastně rozšiřuje. Jedná se o trojúrovňový, bipolární linkový kód bez návratu k nule. AMI rozšiřuje přidáním takzvaných violation bitů při sekvencích čtyř a více nul. Kódování je jinak stejné jako u AMI, takže nula se kóduje jako nula a jednička střídavě jako kladná a záporná úroveň. Sekvence čtyř nul se potom kóduje jako 000V (V - violation) a pokud na vstup přivedeme nul osm, tak se přidává na začátek každé čtveřice violation bit B - tedy výsledný kód je B00V B00V. Violation bity se pak interpretují jako jednička, ale mají stejnou úroveň jako poslední předchozí jednička. Díky tomu má HDB3 lepší synchronizaci než AMI, a proto jej taky nahradil společně s dalšími kódy odvozených z AMI.[8]

3.5 MLT-3

MLT-3 je trojúrovňový, bipolární linkový kód bez návratu k nule, byl využívaný v technologii FDDI a později v technologii fast ethernet, jelikož se více hodí pro metalická vedení. Kódování spočívá v cyklickém opakování všech tří úrovní při kódování jedničky. Pokud přivedeme na vstup nulu, tak výstupní signál má stejnou úroveň jako předchozí bit. Jeho hlavními výhodami oproti jiným dvojúrovňovým a trojúrovňovým kódům je potřeba menší šířky pásma pro stejnou přenosovou rychlost jako například AMI nebo Manchester. Nevýhodou kódování MLT-3 je však přítomnost stejnosměrné složky signálu. Tento linkový kód byl využíván v technologii FDDI.[9]

3.6 2B1Q

2B1Q je čtyřúrovňový, polární linkový kód bez návratu k nule. Narozdíl od předchozích linkových kódů, které kódovaly jednotlivé bity, 2B1Q kóduje dvojici bitů do jedné úrovně. I když se nejedná o modulaci signálu, technika kódování je velice podobná QPSK modulaci (ta využívá jeden symbol pro dva bity) a je využívána technologií ISDN. Hlavní výhodou oproti předchozím je tedy vyšší přenosová rychlost. Jak se kódují jednotlivé dvojice bitů, můžete vidět na následující tabulce[10] (výstupní hodnoty odpovídají úrovním použitých v ISDN, mohou být tedy i jiné):

Tabulka 1: Kódování 2B1Q

Vstup	Výstup
00	-3V
01	-1V
10	+3V
11	+1V

3.7 PST

PST je trojúrovňový, bipolární linkový kód bez návratu k nule. Stejně jako 2B1Q kóduje vždy dvojici bitů, avšak kóduje je do dvojice úrovní. Je závislý na dvou stavech, podle kterých vybírá dvojici úrovní (jedná se tedy o automat s konečným počtem stavů). Pro dvojice bitů 00 a 11 je výsledná dvojice úrovní vždy stejná, ale pro 01 a 10 se rozhoduje na základě stavu, ve kterém se nachází. Přechod mezi stavy zajišťují právě dvojice 00 a 11, kdy po přivedení jedné z nich na vstup zároveň přejde do jiného stavu.[11] Jak se kódují jednotlivé dvojice bitů, můžete vidět na následující tabulce:

Tabulka 2: Kódování PST

Vstup	Výstup
00	- +
01	0 + nebo 0 -
10	+ 0 nebo - 0
11	+ -

3.8 4B5B

4B5B je dvojúrovňový, unipolární linkový kód bez návratu k nule. 4B5B kóduje blok čtyř bitů do bloku pěti bitů. Výsledná pětice bitů je jasně nadefinována a můžete ji vidět v následující tabulce:[8]

Tabulka 3: Kódování 4B5B

Vstup	Výstup	Vstup	Výstup
0000	11110	1000	10010
0001	01001	1001	10011
0010	10100	1010	10110
0011	10101	1011	10111
0100	01010	1100	11010
0101	01011	1101	11011
0110	01110	1110	11100
0111	01111	1111	11101

4 Animace

V předchozích kapitolách jsem popsal linkové kódy a jak se pracuje v prostředí LaTeXu s využitím maker PGF a TikZ, nyní se pustím do vytváření samotných animací. Animací bude celkem osm a jenotlivé animace budou ukazovat princip linkových kódů ze 3. kapitoly. Každý dokument s animací bude typu article, tedy článek. Na začátku každé animace musí být samozřejmě uvedené balíčky, které budu v animaci využívat. Kostra každé animace bude tedy vypadat následovně:

```
\documentclass[]{article}

\usepackage{animate}
\usepackage{tikz}
\usepackage{pgfplots}
\usepackage{pgfplotstable}
\usetikzlibrary{calc}

\begin{document}

\end{document}
```

Výpis 9: Kostra každé animace

K čemu mi budou první tři balíčky animate, tikz a pgfplots je jasné z prvních dvou kapitol bakalářské práce. Balíček pgfplotstable je nutný pro předávání souřadnic jednotlivým grafům, které jsou uloženy v textovém souboru (souřadnice jsou rozděleny do sloupců a každý sloupec začíná označením, co dané souřadnice reprezentují). Posledním příkazem pro import `\usetikzlibrary` se naimportuje knihovna balíčku TikZ calc, kterou budu potřebovat v grafech pro výpočet zadaných souřadnic. Knihovna totiž obsahuje základní matematické funkce, které se jinak do dokumentu nenaimportují.

Každá animace bude obsahovat dva pod sebou umístěné grafy. První graf bude obsahovat vstupní data a druhý graf zakódované vstupní bity podle daného linkového kódu. V prostředí pro nastavení os si pak nadefinuji vlastnosti vzhledu grafu. Grafy neobsahují osu y, protože ta není k pochopení principu linkových kódů důležitá. Osa x pak neobsahuje číselný popis ze stejného důvodu, navíc je animace pak přehlednější. Celý proces se děje v čase (nejčastěji v desítkách milisekund), přesto je příjem a kódování časováno clock signálem, což je další důvod absence popisu osy x. Prostředí grafů v animacích bude tedy vypadat následovně:

```
\begin{tikzpicture}

   \begin{axis}[
   name = input, axis lines = center, axis y line=none,
   width = 15cm, height = 5cm,
   xmin = 0, xmax = 11, ymin = 0, ymax = 1.2,
   xmajorgrids = true, xticklabels = {,,},
   xlabel = {t [ms]}, title = {Vstupní data}]
   \end{axis}

   \begin{axis}[
   name = output, axis lines = center, axis y line=none,
   at = {($(input.south)-(0,1cm)$)}, anchor = north,
   width = 15cm, height = 7.5cm,
   xmin = 0, xmax = 11, ymin = -1.2, ymax = 1.2,
   xmajorgrids = true, xticklabels = {,,},
   xlabel = {t [ms]}, title = {Linkový kód}]
   \end{axis}

\end{tikzpicture}
```

Výpis 10: Nastavení vlastností vzhledu os

Přeložený kód do formátu PDF vypadá takto:

Většina příkazů v poli pro vlastnosti grafu je jasně popsána v 2. kapitole. V následujícím přehledu uvádím tudíž jen ty, kterými jsem se v ní nezabýval:

- axis lines
  - příkaz udává pozici os (defaultní vzhled, jsou osy do čtverce)
  - center znamená, že osy se protnou v nule
  - top by znamenalo, že se osy protnou v maximech atd.
- axis y line
  - nastavení osy y, v mém případě není žádná, tedy none
- at
  - určuje pozici grafu
  - zápis $((clk.south) - (0, 1cm))$ znamená, že graf bude umístěn pod grafem clk a bude mezi nimi mezera 1 centimetr
- xmajorgrids

  - určuje, zda-li bude na majoritních bodech osy x mřížka nebo nebude

- xticklabels
  - určuje hodnoty uvedené na ose x
  - zápis „ znamená, že na ose x nebudou vypsány žádné hodnoty

Vlastní animaci pak udělám pomocí příkazu \multiframe v prostředí animateinline, kdy každá animace obsahuje 102 snímků, kromě animace linkového kódu 4B5B, ten jich obsahuje 142. Více snímků obsahuje proto, že v animaci využívám dvě proměnné (druhá proměnná mi simuluje zpoždění druhého grafu). Všechen obsah tikzpicture jsem potom vkládal do tohoto příkazu. Rychlost vykreslování snímků jsem nastavil na hodnotu 8 snímků za sekundu. Úvodním snímkem je poslední snímek a všechny animace obsahují uživatelské rozhraní pro řízení animace.

```
\begin{animateinline}[poster = last, controls]{8}

   \multiframe{102}{Rx=-0.1+0.1}{
      \begin{tikzpicture}
         %Kód pro vykreslení grafů
      \end{tikzpicture}
   }
\end{animateinline}
```

Výpis 11: Použití prostředí animateinline a příkazu multiframe

Vstupem bude 10 bitů a tomu budou odpovídat zakódována data. Souřadnice předám grafu pomocí textového souboru coord.dat, který je vytvořen na začátku animace. To mě však přivedlo k prvnímu problému, se kterým jsem se při vytváření animací setkal. Uvažoval jsem, jakým způsobem budu reprezentovat data pro jednotlivé grafy v každé animaci. Vstupem každého linkového kódu je posloupnost bitů, které se následně podle nějakého pravidla kódují. Data jsou ovšem nahodilá a nelze je proto popsat žádnou matematickou funkcí. Jelikož výstup každého linkového kódu jistým způsobem odpovídá vstupním bitům, tak je taktéž nelze definovat žádnou matematickou funkcí. Jedinou možností tím pádem bylo nadefinovat si všechny souřadnice sám. Jelikož by to byla velmi zdlouhavá práce, tak jsem využil volně dostupný software GNU Octave a souřadnice si pomocí něj zčásti vygeneroval.

```
\begin{filecontents}{coord.dat}
   X        Yi  Yo
   0        0   0
   0        1   1
   0.1      1   1
   0.2      1   1
   .        .   .
   .        .   .
   .        .   .
   10       0   0
\end{filecontents}
```

Výpis 12: Ukázka souboru coord.dat

Soubor coord.dat obsahuje každá animace a je vždy rozdělen do 3 sloupců:

- 1. sloupec 'X' - obsahuje všechny body x-ové osy od 0 do 10 s krokem 0,1 (některé jsou zdvojené z důvodu přechodu z 0 na 1 nebo naopak)
- 2. sloupec 'Yi' - obsahuje hodnoty, které nabývají jednotlivé vstupní bity, tedy hodnoty osy y (index 'i' značí input)
- 3. sloupec 'Yo' - obsahuje hodnoty osy y pro výstupní zakódované bity podle daného linkového kódu (index 'o' značí output)

Důvodem tak obsáhlého textového souboru se souřadnicemi, je způsob animování každého grafu. Samotný příkaz `\multiframe` nezajišťuje animaci grafu, tu jsem zajistil příkazem `restrict ~x~to~domain`. Pokud jej totiž uvedu do kódu grafu, tak omezí část grafu, která bude vykreslena. V případě, že bych v jedné nebo více částech osy x vynechal dané souřadnice, projevilo by se to v animaci skokem od poslední souřadnice k další zadané souřadnici (tzn. nezadaná část by byla vynechána a poté by se vykreslila najednou). Každý nový snímek tudíž vykreslí znovu všechny části v dané doméně. Souřadnice jsou pak grafu předány příkazem `table[]{}`, kdy do hranatých závorek je nutné specifikovat, jaký sloupec z textového souboru přiřadit jaké ose. Ve složených závorkách je potom název souboru. Je důležité nezapomenout za složenými závorkami středník, i když jej v TEXovském zdrojovém kódu příliš často nevyužívám. V tomto případě by to bez něj nefungovalo.

Pro přehlednost vstupních bitů grafů 'Data' jsem do animace přidal vykreslování hodnoty každého bitu. Ty se vykreslí do středu každého pulsu jako bod grafu pomocí příkazu `\node []`. Za ním se pomocí syntaxe pro určování pozice zadají přímo souřadnice v grafu, kde se daný bod má vykreslit. Jelikož jsem nechtěl, aby již na začátku celé animace byly vykresleny jednotlivé hodnoty bitů, nadefinoval jsem na začátku nový counter a poté jsem pomocí podmínek

\ifthenelse zajistil, aby se vykreslily právě po doběhnutí každého pulsu. Counter jsem si musel vytvořit z toho důvodu, že proměnná, kterou si vytvořím v rámci příkazu \multiframe, nemůže vstupovat do podmínek. Podmínky z balíčku ifthenelse operují pouze s integer proměnnými a s jinými typy si nedokáží poradit. Ukázku zajištění vykreslení hodnoty jednotlivých bitů můžeme vidět na následujícím kódu:

```
\ifthenelse{\thenodes > 11}{
\node[] at (axis cs: 0.5, 0.5) {1};}{}
\ifthenelse{\thenodes > 21}{
\node[] at (axis cs: 1.5, 0.5) {0};}{}
\ifthenelse{\thenodes > 31}{
\node[] at (axis cs: 2.5, 0.5) {1};}{}
\ifthenelse{\thenodes > 41}{
\node[] at (axis cs: 3.5, 0.5) {1};}{}
\ifthenelse{\thenodes > 51}{
\node[] at (axis cs: 4.5, 0.5) {0};}{}
\ifthenelse{\thenodes > 61}{
\node[] at (axis cs: 5.5, 0.5) {0};}{}
\ifthenelse{\thenodes > 71}{
\node[] at (axis cs: 6.5, 0.5) {1};}{}
\ifthenelse{\thenodes > 81}{
\node[] at (axis cs: 7.5, 0.5) {1};}{}
\ifthenelse{\thenodes > 91}{
\node[] at (axis cs: 8.5, 0.5) {0};}{}
\ifthenelse{\thenodes > 101}{
\node[] at (axis cs: 9.5, 0.5) {1};}{}
```

Výpis 13: Ukázka vykreslení hodnoty jednotlivých bitů

Dalším atributem animací je barevné vyplnění všech pulzů v grafech pro data a zakódovana data. To jsem udělal, aby byla ukázka principu linkového kódu přehlednější. K vybarvení dojde až na konci celé animace, což jsem zajistil pomocí příkazu pro podmínku \ifthenelse. Do podmínky vstupuje stejný counter, který jsem vytvořil pro vykreslené hodnoty bitů. Není to však atributem všech animací. U animací linkových kódů Manchester a diferenciální Manchester, se takhle vybarví jen graf pro data. Pokud bych tenhle atribut přidal i grafům pro zakódovaná data, tak by animace byla poněkud nepřehledná. U těchto dvou linkových kódů, jak bylo popsáno v 3. kapitole, se signál v průběhu kódování jednoho bitu vrací k nule a vybarvení by mohlo být matoucí.

```
\ifthenelse{\thenodes > 101}{
\addplot[thick, const plot, fill = yellow, draw = red] table[x = X, y = Yi]{
    coord.dat};}{}
```

Výpis 14: Ukázka vybarvení grafů

Jak je z kódu viditelné, graf se vybarví žlutou barvou a znovu se obtáhne červenou. Červenou barvou se musí obtáhnout znovu, protože pokud by to nebylo vedeno, graf by se automaticky obtáhnul barvou černou. To by sice příliš nevadilo, ale v průběhu animace se graf vykresluje červenou barvou, proto jsem něchtěl, aby se po animaci barva změnila.

4.1 Rozdíly animací

Výše popsané vlastnosti animací se vztahují ke všem animacím, avšak u některých z nich bylo potřeba je trochu pozměnit. Týká se to hlavně animací linkových kódů 2B1Q, PST a 4B5B. Tyhle tři linkové kódy totiž kódují více bitů najednou, proto jsem musel zajistit, aby to bylo z animací zřejmé. Z tohoto důvodu dané animace obsahují dvě proměnné v příkazu `\multiframe` . Druhá vytvořená proměnná simuluje zpoždění vykreslování druhého grafu, aby bylo jasné, že je zároveň kódováno více bitů, a přitom záleží na hodnotě obou. Linkové kódy 2B1Q a PST kódují najednou dva bity, každá dvojice kódovaných bitů je pak vybarvena jinou barvou. Při vybarvování grafu více než jednou barvou, je třeba za každý příkaz pro vykreslení grafu přidat upřesňující příkaz `\closedcycle`, jinak by se barvy různě promíchaly a graf by byl nepřehledný. K vybarvení dojde vždy po doběhnutí každého přeneseného symbolu.

```
\ifthenelse{\thenodes > 51}{
\addplot[thick, fill = yellow, draw = red, restrict x to domain=0:1]
   table[x = X, y = Yo]{coord.dat} \closedcycle;}{}

\ifthenelse{\thenodes > 61}{
\addplot[thick, fill = green, draw = red, restrict x to domain=1:2]
   table[x = X, y = Yo]{coord.dat} \closedcycle;}{}

\ifthenelse{\thenodes > 71}{
\addplot[thick, fill = blue, draw = red, restrict x to domain=2:3]
   table[x = X, y = Yo]{coord.dat} \closedcycle;}{}
```

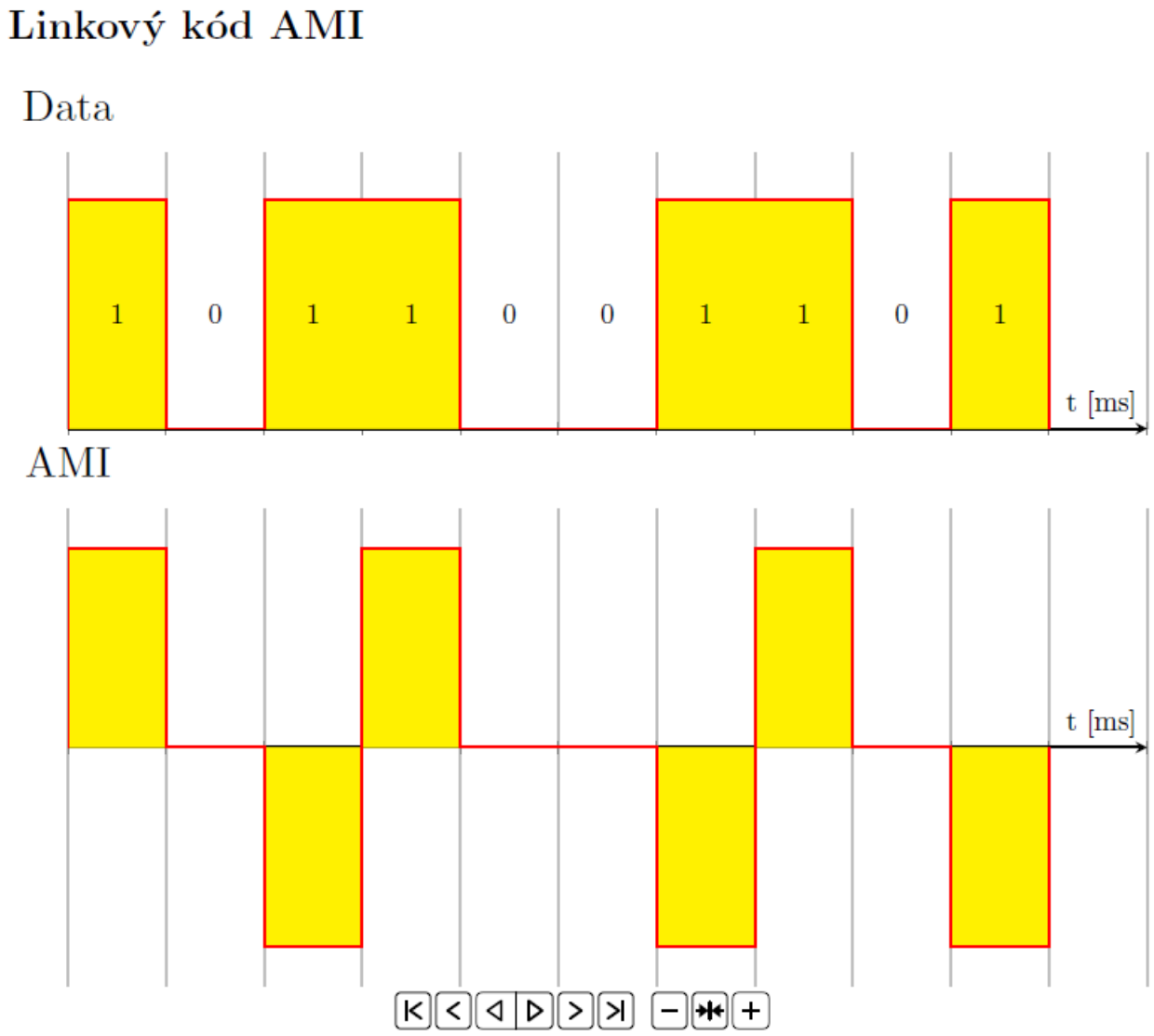
Výpis 15: Ukázka vybarvení grafu více barvami

U linkového kódu 4B5B se první čtyři bity dat a pět bitů zakódovaných dat vybarví žlutou barvou již potom, co se úplně vykreslí. Zbytek je pak vybarven zelenou. Díky tomu by mělo být jasné, jaká posloupnost bitů se zakódovala do jaké posloupnosti. Navíc je u této animace vykreslena hodnota výstupních zakódovaných dat. Linkový kód 4B5B, jak už je z názvu patrné,

kóduje čtveřici bitů do pětice bitů. U ostatních linkových kódů se výstupní hodnoty mohou lišit na základě použité technologie, avšak pokud je výstupem bit, tak může nabývat hodnot jen 0 a 1.

4.2 Ukázka animace

V následující podkapitole uvádím ukázku výstupu zdrojového kódu v PDF formátu linkového kódu AMI, jak jsem popisoval výše. Jedná se však jen o obrázek, tudíž animace není spustitelná. Veškeré zdrojové kódy a jejich výstup v soboru PDF naleznete v přílohách bakalářské práce.



Obrázek 8: Výsledná animace linkového kódu AMI

5 Začlenění animací do souboru PDF

Všechny vytvořené animace pomocí balíčku animate lze začlenit do formátu PDF. V mém případě jsou animace vytvořeny navíc pomocí balíčků PGF a TikZ, mohou být ale vytvořeny taktéž například ze sady po sobě jdoucích obrázků. Animace jsou dále řízeny JavaScriptem, kvůli tomu je nelze přehrát pomocí jakéhokoliv PDF prohlížeče. Například, když jsem se pokoušel animace spustit pomocí webového prohlížeče, tak jsem samozřejmě neuspěl. Ovšem na starších verzích webových prohlížečů, bylo možné nainstalovat Adobe Reader plugin, díky kterému je možné animace v prohlížeči spustit (animace by byly v tomhle případě však velmi pomalé). V následujícím krátkém přehledu uvádím v jakých PDF prohlížečích je možné animace spustit:

- Acrobat Reader od firmy Adobe
  - všechny animace jsem odzkoušel a bylo je možné úspěšně spustit (využíval jsem edici Acrobat Reader DC), animace by měly fungovat od verze 7
- PDF-XChange Viewer od firmy Tracker Software Products
  - jedná se o komerční software a je nutné mít zakoupenou licenci, existuje však i bezplatná trial verze, díky které jsem tento software mohl otestovat
  - animace je možné spusti jen v případě, že animace obsahuje možnost 'method=widget' nebo 'method=ocg', bez jednoho z těchto příkazů nebude animace v daném PDF prohlížeči fungovat
- Foxit Reader od firmy Foxit
  - taktéž se jedná o komerční software, s možností bezplatné trial verze. Animace v tomto softwaru fungovaly taktéž bez problémů

Při snaze začlenit animace do této práce, jsem však narazil na jeden problém. Jde o to, že vytvořené animace nemusí fungovat ve formátu PDF/A. V prohlížeči Acrobat Reader se od nějaké verze při otvírání PDF/A souboru s animací zeptá, zda-li povolíme úpravy v dokumentu, pak by animace měla fungovat. Pro překlad zdrojového kódu animací jsem používal TeXovský engine pdfLaTeX. Vyzkoušel jsem také další dva enginy a to LuaLaTeX a XeLaTeX, s prvním zmiňovaným nebyl žádný problém a animace byly normálně spustitelné. Ovšem při překladu pomocí XeLaTeX-u, došlo k přečerpání vyhrazené paměti. Tento problém jsem však vyřešil úpravou konfiguračního souboru pro XeLaTeX, do kterého jsem přidal příkaz `main_memory=12000000` a poté jsem službu restartoval. Zdrojový kód animace šel následně přeložit bez problému a animace také fungovala bez problému. Jedná se také o chybu, která může nastat a je zmíněna v popisu balíčku animate. Další možné chyby jsou například:

- animace nefungují v případě, že je PDF vytvořeno s verzí Ghostscriptu starší než 9.15

- animace nemusí fungovat v případě, že PDF dokument byl rozdělen do více souborů
- animace nefungují v případě, že byly přidány do jiného dokumentu pomocí příkazů `\includegraphics` nebo `\includepdf` (animace se pak jeví jen jako obrázek)[6]

Závěr

Hlavním přínosem mé bakalářské práce je vytvořená sada animací, ze kterých je snadno pochopitelné, jak dané linkové kódy fungují. Veškeré vytvořené animace můžete vidět v elektronických přílohách práce. V textu uvádím postup, jakým jsem se řídil a který jsem aplikoval u všech animací. Následně uvádím ukázku vzhledu jedné vytvořené animace, a to linkového kódu AMI. Při vytváření jsem se setkal s několika problémy, avšak všechny se mi povedlo následně vyřešit. Prvním problémem bylo, z jakých dat vykreslovat grafy do animací. Vstupní ani výstupní data nelze popsat žádnou matematickou funkcí a pokud by dané grafy obsahovaly jen minimum potřebných souřadnic, tak by animace byla kouskována. Proto jsem využil volně dostupného softwaru GNU Octave a souřadnice na x-ové ose jsem si rozdělil do sto bodů s krokem 0,1. Následně jsem každému intervalu hodnot přiřadil jejich funkční hodnotu a aby nebyly v kódu u každého grafu tyhle souřadnice uvedeny, tak jsem si na začátku vytvořil textový soubor coord.dat, který mi tato data uchoval. Další problém se vyskytl při vytváření animace linkového kódu 4B5B, a to, když jsem se snažil docílit toho, aby byl graf vybarven dvěma barvami a bylo tak lehce rozeznatelné, které čtyři bity se zakódovaly do jaké pětice. Balíček TikZ totiž nerozeznal, kdy končí jedna část vybarvená žlutou barvou a kdy začíná část se zelenou barvou. Za definicí každého grafu jsem tak musel vložit příkaz pro uzavřenou část grafu a až potom se graf vybarvil správně. Výsledné animace mohou sloužit jako výukový materiál, a to hlavně v předmětech Přenosové systémy a média nebo Přenos dat.

Dalším přínosem bakalářské práce je vytvořený manuál týkající se prostředí LaTeXu s hlavním zaměření na vytváření animací a s animacemi spjatá rešerše o principu linkových kódů. Ve zmíněném stručném manuálu kladu důraz hlavně na práci s balíčkem animate a balíčky PGF a TikZ. Tyto balíčky jsou určeny k vytváření vektorové grafiky a jsou tak ideálním pomocníkem pro vytváření například grafů, schémat elektrických obvodů nebo různých diagramů. Po přečtení práce, by začínající uživatel LaTeXu neměl mít problém s těmito balíčky pracovat. Hlavní důraz je pak kladen na vytváření animací spustitelných v PDF souborech, kdy se animují právě grafy vytvořené pomocí balíčku PGF, respektive jeho podbalíčku pgfplots. Pro vytváření animací využívám dalších balíčků LaTeXu, a to zejména balíčky animate, ifthen a forloop. V bakalářské práci popisuji hned dva způsoby, jak si vytvořit svou animaci. Vytvořené animace mohou obsahovat uživatelský interface, pomocí kterého lze animaci ovládat. Obsahem rešerše je navíc popis hlavních principů linkových kódů, které jsou využívany v přenosové technologii. V práci uvádím jejich obecný popis, hlavní rozdělení a následně jsem osm z nich v krátkosti popsal. A právě pro těchto osm linkových kódů vytvářím v praktické části animace. Jedná se o linkové kódy AMI, Manchester, Diferenciální Manchester, HDB3, MLT-3, PST, 2B1Q a 4B5B.

Posledním přínosem bakalářské práce je pak začlenění animací do souboru s formátem PDF a otestování možností překladů a PDF prohlížečů. Kvůli tomu je však nutné zdrojové kódy přeložit celkem dvakrát, s prvním přeložením se vytvoří grafická stránka animace a druhým přeložením se zpřístupní uživatelský interface. Všechny animace jsou exportovány ve formátu PDF, avšak jsou

řízeny Javascriptem, proto není možné je spustit jakýmkoliv PDF prohlížečem. Podporované a mnou odzkoušené jsou Acrobat Reader, PDF-XChange Viewer a Foxit Reader. V případě PDF-XChange Vieweru je třeba do vlastností animace navíc vložit upřesnění pro metodu animace. Po celou dobu jsem využíval TeXovský software Miktex 2.9 pro operační systém windows. Taktéž jsem vyzkoušel tři enginy pro překládání zdrojového kódu a to pdfLaTeX, LuaLaTeX a XeLaTeX. To však na funkčnost animací nemělo žádny vliv.

Literatura

[1] *LaTeX for Beginners [online].* Edinburgh, 2014, 31 s. Edition 5. 3722-2014. Dostupné také z: http://www.docs.is.ed.ac.uk/skills/documents/3722/3722-2014.pdf. Workbook. The University of Edinburgh.

[2] MACHÁČEK, Pavel. *Animované prezentace analogových a digitálních modulací v prostředí PGF/TikZ [online].* Ostrava, 2016, 38 s. Dostupné také z: http://hdl.handle.net/10084/116089. Bakalářská práce. Vysoká škola báňská - Technická univerzita Ostrava. Fakulta elektrotechniky a informatiky. Vedoucí práce Ing. Jan Skapa, Ph.D.

[3] TANTAU, Till. *The TikZ and PGF Packages [online].* Lübeck, 2012, 880 s. Dostupné také z: http://www.texample.net/media/pgf/builds/pgfmanualCVS2012-11-04.pdf. Manuál. Universität zu Lübeck. Institut für Theoretische Informatik.

[4] FEUERSÄNGER, Christian. *Manual for Package pgfplots [online].* 2018. Dostupné z: http://ftp.cvut.cz/tex-archive/graphics/pgf/contrib/pgfplots/doc/pgfplots.pdf. Manuál.

[5] CARLISLE, David. *The ifthen package [online].* 2014. Dostupné z: http://ftp.cvut.cz/tex-archive/macros/latex/base/ifthen.pdf. Manuál.

[6] GRAHN, Alexander. *The animate Package [online].* 2016, 26 s. Dostupné také z: http://mirrors.concertpass.com/tex-archive/macros/latex/contrib/animate/animate.pdf. Manuál.

[7] JAIN, Kartik a GOEL, Ankit. *A new line code based on overlapping of block of bits [online].* In: . Delhi, India, 2013. ISBN 978-1-4799-1085-4. Dostupné z: https://ieeexplore.ieee.org/document/6749442

[8] FAIRHURST, Gorry. *Internet Communications Engineering - A Tutorial. [online].* 2001. Dostupné z: https://erg.abdn.ac.uk/users/gorry/eg3561/phy-pages/

[9] MAZZOLA, Mario; CAFIERO, Luca a DENICOLO, Maurilio. *Method and apparatus for multilevel encoding for a local area network. : United States Patent [online].* Spojené Státy Americké, 1994. Dostupné z: https://patents.google.com/patent/US5280500

[10] *ISDN 2B1Q signal format. RF Wireless World [online].* Dostupné z: http://www.rfwireless-world.com/Tutorials/ISDN-2B1Q-signal-format.html

[11] SCHELLEKENS, Thomas. *Ternary Line Codes and their Efficiency [online].* Twente, Nizozemsko, 2007. Dostupné také z: https://www.utwente.nl/en/eemcs/dacs/assignments/completed/bachelor/reports/B-assignment_Schellekens.pdf. Bakalářská práce. University of Twente, the Netherlands.